We are going to discuss our design changes that we would make in order to implement Key Trick B.

1. Create a boss bullet component and add this “flag” to all bullets fired by the boss
2. Create a system to check if a boss bullet hitbox is touching an enemy bullet hitbox
    a. Have the enemy bullet copy the velocity vector of the boss bullet (direction and speed)
3. Create a system to check if any bullets have touched the boundaries of the map
    a. When a bullet touch the boundary, have them despawn
        i. If the bullet despawning is a boss bullet then spawn new bullets that fire inwards

Because our ECS architecture uses components and systems, we need to add three main components: (1) Boss bullet component (2) Boss bullet hitbox system (3) Bullet boundary system.

(1) Boss Bullet Component:

This will be a simple component that will act like a flag for our systems. When the boss spawns the special “boss bullets”, we then add this component to them. These bullets will be the same to every other bullet sprite on screen otherwise (contains a moveComponent, a CollisionComponent, a animationComponent, a HitboxComponent, etc.).

(2) Boss Bullet Hitbox System:

This will be the system that handles the logic for hitbox detection and also updating the enemy bullets to copy the boss bullets velocity vector. This system will need some ECS attributes, that specify which entities this system should interact with. In this case, we need to specify that this system is interacting with all enemy bullets. These bullets then check if their hitbox has collided with any boss bullets which can be retrieved using the enetity.Get<BossBulletComponent>() function call (this is where the Boss Bullet Component flag comes in use). This system will be repeatedly called every tick of our game in our Interactive State.

From here, we can write the logic for each bullet that does collide with a boss bullet. We will write function calls on the bullet (entity.updateVelocity(Vector2 velocity) for example). This will simulate what we see in the video where the boss bullet carries these smaller bullets to the edge of the screen.

(3) Bullet Boundary System:

This will be the last system needed. We already have a system that deals with removing bullets when their hitboxes move past the boundaries of the map. We should either update this system, or create a new system specifically for the boss bullet. In this new case, when the boss bullet despawns, we then want to spawn a cluster of new bullets. These bullets will then be assigned

a velocity vector that points somewhere towards the middle of the screen. This will be the last step of simulating Key Trick B.